WHAT WE CLAIM IS:

1. A method for improving quality-of-service in a distributed computing system, said system including a multiplicity of network-connected worker processors and at least one supervisory processor, said supervisory processor configured to assign tasks to said worker processors, said method comprising:

identifying one or more of said tasks as critical task(s);

assigning each of said tasks, including said critical task(s), to a worker processor;

redundantly assigning each of said one or more critical task(s) to a worker processor; and,

monitoring the status of said assigned tasks to determine when all of said tasks have been completed by at least one worker processor.

2. A method for improving quality-of-service in a distributed computing system, as defined in claim 1, further comprising:

monitoring, on a substantially continuous basis, the status of at least the worker processor(s) that have been assigned the non-critical task(s).

3. A method for improving quality-of-service in a distributed computing system, as defined in claim 2, wherein monitoring, on a substantially continuous basis, the status of at least the worker processor(s) that have been assigned the

non-critical task(s) comprises receiving status messages from at least each of the worker processor(s) that have been assigned non-critical task(s) until each said processor completes its assigned task.

4. A method for improving quality-of-service in a distributed computing system, as defined in claim 2, wherein monitoring, on a substantially continuous basis, the status of at least the worker processor(s) that have been assigned the non-critical task(s) comprises detecting abnormalities in the operation of the worker processor(s) that have been assigned non-critical task(s), and/or their associated network connections, by detecting an absence of expected status message(s) received by said at least one supervisory processor.

5. A method for improving quality-of-service in a distributed computing system, as defined in claim 4, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once every ten minutes.

6. A method for improving quality-of-service in a distributed computing system, as defined in claim 4, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once each minute.

7. A method for improving quality-of-service in a distributed computing system, as defined in claim 4, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated

at least once each second.

8. A method for improving quality-of-service in a distributed computing system, as defined in claim 4, wherein said act of detecting an absence of expected status message(s) received by said at least one supervisory processor is repeated at least once every tenth of a second.

9. A method for improving quality-of-service in a distributed computing system, as defined in claim 2, wherein monitoring, on a substantially continuous basis, the status of at least the worker processor(s) that have been assigned the non-critical task(s) comprises:

detecting the presence of non-assigned-task-related activity on at least said worker processor(s) that have been assigned the non-critical task(s).

10. A method for improving quality-of-service in a distributed computing system, as defined in claim 9, wherein detecting the presence of non-assigned-task-related activity includes:

running an activity monitor program on at least each of the worker processor(s) that have been assigned non-critical task(s).

11. A method for improving quality-of-service in a distributed computing system, as defined in claim 10, wherein:

the activity monitor programs behave substantially like screen saver programs.

12. A method for improving quality-of-service in a distributed computing system, as defined in claim 10, wherein:

the activity monitory programs send, in response to detection of keyboard activity, a message to at least one of said at least one supervisory processor(s).

13. A method for improving quality-of-service in a distributed computing system, as defined in claim 10, wherein:

the activity monitory programs send, in response to detection of mouse activity, a message to at least one of said at least one supervisory processor(s).

14. A method for improving quality-of-service in a distributed computing system, as defined in claim 10, wherein:

the activity monitory programs send, in response to detection of pointer activity, a message to at least one of said at least one supervisory processor(s).

15. A method for improving quality-of-service in a distributed computing system, as defined in claim 10, wherein:

the activity monitory programs send, in response to detection of touchscreen activity, a message to at least one of said at least one supervisory processor(s).

16. A method for improving quality-of-service in a distributed computing system, as defined in claim 10, wherein:

the activity monitory programs send, in response to detection of voice activity, a message to at least one of said at least one supervisory processor(s).

17. A method for improving quality-of-service in a distributed computing system, as defined in claim 10, wherein:

the activity monitory programs send, in response to detection of execution of substantial non-assigned-task-related processes, a message to at least one of said at least one supervisory processor(s).

18. A method for improving quality-of-service in a distributed computing system, as defined in claim 9, wherein detecting the presence of non-assigned-task-related activity includes:

determining, in response to an activity monitor message received by at least one of said at least one supervisory of said processor(s), that at least one of said worker processors is undertaking non-assigned-task-related activity.

19. A method for improving quality-of-service in a distributed computing system, as defined in claim 18, wherein the activity monitor message is generated by an activity monitor program running on one of said assigned worker processors.

20. A method for operating a peer-to-peer distributed computing system, comprising:

providing a pool of worker processors, each having installed worker processor software, and each connected to an always-on, peer-to-peer computer network;

providing at least one supervisory processor, also connected to said always-on, peer-to-peer computer network;

using said at least one supervisory processor to monitor the status of worker processors expected to be engaged in the processing of assigned tasks; and,

using said at least one supervisory processor to redundantly assign one more critical task(s) to one or more additional worker processors.

21. A method for operating a peer-to-peer distributed computing system, as defined in claim 20, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network through a high-bandwidth connection.

22. A method for operating a peer-to-peer distributed computing system, as defined in claim 21, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 100 kilobits/sec.

23. A method for operating a peer-to-peer distributed computing system, as defined in claim 21, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 250 kilobits/sec.

24. A method for operating a peer-to-peer distributed computing system, as defined in claim 21, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 1 megabit/sec.

25. A method for operating a peer-to-peer distributed computing system, as defined in claim 21, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 10 megabits/sec.

26. A method for operating a peer-to-peer distributed computing system, as defined in claim 21, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 100 megabits/sec.

27. A method for operating a peer-to-peer distributed computing system, as defined in claim 21, wherein providing a pool of worker processors further includes ensuring that each of said worker processors is linked to said always-on, peer-to-peer computer network at a data rate of at least 1 gigabit/sec.

28. A method for operating a peer-to-peer distributed computing system, as defined in claim 20, wherein using said at least one supervisory processor to monitor the status of worker processors expected to be engaged in the processing of assigned tasks includes:

sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks.

29. A method for operating a peer-to-peer distributed computing system, as defined in claim 28, wherein said process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least once every second.

30. A method for operating a peer-to-peer distributed computing system, as defined in claim 28, wherein said process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least once every tenth of a second.

31. A method for operating a peer-to-peer distributed computing system, as defined in claim 28, wherein said process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least

once every hundredth of a second.

32. A method for operating a peer-to-peer distributed computing system, as defined in claim 28, wherein said process of sending a status-request message to, and receiving a return acknowledgement from, each worker processor that is expected to be engaged in the processing of assigned tasks is repeated at least once every millisecond.

33. A method for operating a peer-to-peer distributed computing system, as defined in claim 20, wherein using said at least one supervisory processor to monitor the status of worker processors expected to be engaged in the processing of assigned tasks includes:

periodically checking to ensure that a heartbeat message has been received, within a preselected frequency interval, from each worker processor that is expected to be engaged in the processing of assigned tasks.

34. A method for operating a peer-to-peer distributed computing system, as defined in claim 33, wherein said preselected frequency interval is less than one second.

35. A method for operating a peer-to-peer distributed computing system, as defined in claim 33, wherein said preselected frequency interval is less than one tenth of a second.

36. A method for operating a peer-to-peer distributed computing system, as defined in claim 33, wherein said preselected frequency interval is less than one hundredth of a second.

37. A method for operating a peer-to-peer distributed computing system, as defined in claim 33, wherein said preselected frequency interval is less than one millisecond.

38. A method for performing a job using a peer-to-peer network-connected distributed computing system, the job comprising a plurality of tasks, the method comprising:

initiating execution of each of said plurality of tasks on a different processor connected to said peer-to-peer computer network;

initiating redundant execution of at least one of said plurality of tasks on yet a different processor connected to said peer-to-peer computer network; and,

once each of said plurality of tasks has been completed by at least one processor, reporting completion of said job via said peer-to-peer computer network.

39. A method for performing a job using a peer-to-peer network-connected distributed computing system, as defined in claim 38, wherein said at least one of said plurality of tasks that is/are redundantly assigned is/are critical task(s).

40. A method for performing a job using a peer-to-peer network-connected distributed computing system, as defined in claim 38, further comprising:

monitoring, on a periodic basis, to ensure that progress is being made toward completion of said job.

41. A method for performing a job using a peer-to-peer network-connected distributed computing system, as defined in claim 40, wherein said monitoring is performed at least once every 10 seconds.

42. A method for performing a job using a peer-to-peer network-connected distributed computing system, as defined in claim 40, wherein said monitoring is performed at least once a second.

43. A method for performing a job using a peer-to-peer network-connected distributed computing system, as defined in claim 40, wherein said monitoring is performed at least once every tenth of a second.

44. A method for performing a job using a peer-to-peer network-connected distributed computing system, as defined in claim 40, wherein said monitoring is performed at least once every hundredth of a second.

45. A method for performing a job using a peer-to-peer network-connected distributed computing system, as defined in claim 40, wherein said monitoring is performed at least once every millisecond.

46. A method for performing a job using a plurality of independent, network-connected processors, the job comprising a plurality of tasks, the method

comprising:

assigning each of said plurality of tasks to a different processor connected to said computer network;

redundantly assigning at least some, but not all, of said plurality of tasks to additional processors connected to said computer network; and,

using said computer network to compile results from the assigned tasks and report completion of the job.

47. A method, as defined in claim 46, wherein redundantly assigning at least some of said plurality of tasks to additional processors comprises assigning critical tasks to additional processors.

48. A method, as defined in claim 46, wherein redundantly assigning at least some of said plurality of tasks to additional processors comprises assigning at least one critical task to at least two additional processors.

49. A method, as defined in claim 46, further comprising:

generating a heartbeat message from each processor executing an assigned task at least once every second.

50. A method, as defined in claim 46, further comprising:

generating a heartbeat message from each processor executing an assigned task at least once every tenth of a second.

51. A method, as defined in claim 46, further comprising:

generating a heartbeat message from each processor executing an assigned task at least once every hundredth of a second.

52. A method, as defined in claim 46, further comprising:

generating a heartbeat message from each processor executing an assigned task at least once every millisecond.

53. A method for performing a job using a pool of network-connected processors, the job comprising a plurality of tasks, the number of processors in the pool greater than the number of tasks in the job, the method comprising:

assigning each of said plurality of tasks to at least one processor in said pool;

redundantly assigning at least some of said plurality of tasks until all, or substantially all, of said processors in said pool have been assigned a task; and,

using said computer network to compile results from the assigned tasks and report completion of the job.

54. A method, as defined in claim 53, wherein redundantly assigning at least some of said plurality of tasks includes redundantly assigning a plurality of critical tasks.

55. A method for using redundancy in a network-based distributed processing system to avoid or mitigate delays from failures and/or slowdowns of individual processing elements, the method comprising:

receiving a job request, from a client, over the network;

processing the job request to determine the number, K, of individual tasks to be assigned to individual network-connected processing elements;

determining a subset, N, of said K tasks whose completion is most critical to the overall completion of the job; and,

assigning each of said K tasks to an individual network-connected processing element; and,

redundantly assigning at least some of the N task(s) in said subset to additional network-connected processing element(s).

56. A method, as defined in claim 55, for using redundancy in a network-based distributed processing system to avoid or mitigate delays from failures and/or slowdowns of individual processing elements, wherein determining the subset, N, of said K tasks whose completion is most critical to the overall completion of the job includes assigning, to the subset, task(s) that must be completed before other task(s) can be commenced.

57. A method, as defined in claim 55, for using redundancy in a network-based distributed processing system to avoid or mitigate delays from failures and/or

slowdowns of individual processing elements, wherein determining the subset, N, of said K tasks whose completion is most critical to the overall completion of the job includes assigning, to the subset, task(s) that supply data to other task(s).

58. A method, as defined in claim 55, for using redundancy in a network-based distributed processing system to avoid or mitigate delays from failures and/or slowdowns of individual processing elements, wherein determining the subset, N, of said K tasks whose completion is most critical to the overall completion of the job includes assigning, to the subset, task(s) that is/are likely to require the largest amount of memory.

59. A method, as defined in claim 55, for using redundancy in a network-based distributed processing system to avoid or mitigate delays from failures and/or slowdowns of individual processing elements, wherein determining the subset, N, of said K tasks whose completion is most critical to the overall completion of the job includes assigning, to the subset, task(s) that is/are likely to require the largest amount of local disk space.

60. A method, as defined in claim 55, for using redundancy in a network-based distributed processing system to avoid or mitigate delays from failures and/or slowdowns of individual processing elements, wherein determining the subset, N, of said K tasks whose completion is most critical to the overall completion of the job includes assigning, to the subset, task(s) that is/are likely to require the largest amount of processor time.

61. A method, as defined in claim 55, for using redundancy in a network-based distributed processing system to avoid or mitigate delays from failures and/or slowdowns of individual processing elements, wherein determining the subset, N, of said K tasks whose completion is most critical to the overall completion of the job includes assigning, to the subset, task(s) that is/are likely to require the largest amount of data communication over the network.

62. A method, as defined in claim 55, for using redundancy in a network-based distributed processing system to avoid or mitigate delays from failures and/or slowdowns of individual processing elements, further comprising:

determining, based on completions of certain of said K tasks and/or N redundant task(s), that sufficient tasks have been completed to compile job results; and,

reporting job results to the client over the network.

63. A method for using a group of network-connected processing elements to process a job, the job comprised of a plurality of tasks, one or more of which are critical tasks, the method comprising:

identifying a one or more higher-capacity processing elements among said group of network-connected processing elements;

assigning at least one critical task to at least one of the identified higher-capacity processing elements;

assigning other tasks to other processing elements such that each task

in said job has been assigned to at least one processing element; and,

communicating results from said assigned tasks over said network.

64. A method for using a group of network-connected processing elements to process a job, as defined in claim 63, wherein identifying a one or more higher-capacity processing elements among said group of network-connected processing elements includes evaluating the processing capacity of processing elements in said group based on their execution of previously-assigned tasks.

65. A method for using a group of network-connected processing elements to process a job, as defined in claim 63, wherein identifying a one or more higher-capacity processing elements among said group of network-connected processing elements includes determining the processing capacity of processing elements in said group through use of assigned benchmark tasks.

66. A method for using a group of network-connected processing elements to process a job, as defined in claim 63, wherein identifying a one or more higher-capacity processing elements among said group of network-connected processing elements includes evaluating hardware configurations of at least a plurality of processing elements in said group.

67. A method for using a group of network-connected processing elements to process a job, as defined in claim 63, further comprising:

ensuring that each critical task in the job is assigned to a higher-

capacity processing element.

68. A method for using a group of network-connected processing elements to process a job, as defined in claim 63, further comprising:

storing the amount of time used by said processing elements to execute the assigned tasks; and,

computing a cost for said job based, at least in part, on said stored task execution times.

69. A method for using a group of network-connected processing elements to process a job, as defined in claim 68, wherein computing a cost for said job based, at least in part, on said stored task execution times includes charging a higher incremental rate for time spent executing tasks on higher-capability processing elements than for time spent executing tasks on other processing elements.

70. A method for using a group of network-connected processing elements to process a job, as defined in claim 68, further comprising:

communicating the computed cost for said job over said network.

71. A distributed computing system comprising:

a multiplicity of worker processors;

at least one supervisory processor, configured to assign tasks to, and monitor the status of, said worker processors, said at least one supervisory processor further configured to assign each critical

task to at least two worker processors;

an always-on, peer-to-peer computer network linking said worker processors and said supervisory processor(s); and,

at least one of said at least one supervisory processor(s) including a monitoring module, which monitors the status of worker processors expected to be executing assigned tasks to ensure that the distributed computing system maintains always-live operation.

72. A distributed computing system, as defined in claim 71, wherein the monitoring module receives status messages from at least each of the worker processors expected to be executing assigned tasks.

73. A distributed computing system, as defined in claim 72, wherein the monitoring module detects abnormalities in the operation of said worker processors expected to be executing assigned tasks, and/or their associated network connections, by detecting an absence of expected status messages received from said worker processors.

74. A distributed computing system, as defined in claim 73, wherein the monitoring module checks for an absence of expected status messages at least once each minute.

75. A distributed computing system, as defined in claim 73, wherein the monitoring module checks for an absence of expected status messages at least

once each second.

76. A distributed computing system, as defined in claim 72, wherein the monitoring module detects the presence of non-assigned-task-related activity on the worker processors expected to be executing assigned tasks.

77. A distributed computing system, as defined in claim 76, further comprising:

activity monitor programs running on each of the worker processors expected to be executing assigned tasks.

78. A distributed computing system, as defined in claim 77, wherein the activity monitor programs comprise screensaver programs.

79. A distributed computing system, as defined in claim 77, wherein the activity monitor programs detect at least one of the following types of non-assigned-task-related activity:

keyboard activity;

mouse activity;

pointer activity;

touchscreen activity;

voice activity; and,

execution of substantial non-assigned-task-related processes.

80. A distributed computing system, as defined in claim 77, wherein the activity monitor programs detect at least two of the following types of non-assigned-

task-related activity:

keyboard activity;

mouse activity;

pointer activity;

touchscreen activity;

voice activity; and,

execution of substantial non-assigned-task-related processes.